

ASYNCHRONOUS ENHANCED SHARED SECRET PROVISIONING PROTOCOL

Field of the Invention

5 The present invention relates to the field of secure network registration processes that allow two devices to register with each other, and more particularly to a registration process where two devices learn each other's identities and establish a shared key that can later be used by the devices to mutually authenticate each other, to generate session encryption keys, to generate credentials, and for other purposes.

10 Background of the Invention

 The protection of information and secrets over a network requires the use of secure methods to add new devices to the network. It is possible to breach network security and gain access to network information and secrets through interfering with the registration process of devices with the network. One method of interfering with the registration of network devices 15 is through interjecting an imposter device into the registration process. If this imposter device can successfully pose as the legitimate device during the registration process, then it is possible for the imposter device to register with the network and masquerade as the legitimate device. As a result, the imposter device can gain access to the information and secrets stored 20 on the network. It is therefore desirable to develop methods and systems that can provide a secure method for registering a device with a network.

 A variety of methods and systems are known to facilitate communications between two devices. One such protocol is the Diffie-Hellman key agreement protocol. The Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed by Diffie and Hellman in 1976 and published in the paper "New Directions in Cryptography."

The protocol allows two users to establish a secret key over an insecure medium without any prior secrets. The protocol has two system parameters p and g . They are both public and may be used by all the users in a system. Parameter p is a prime number and parameter g (usually called a generator) is an integer less than p , which is capable of generating every element from

5 1 to $p-1$ when multiplied by itself a certain number of times, modulo the prime p . The protocol depends on the discrete logarithm problem for its security. It assumes that it is computationally infeasible to calculate the shared secret key $k=g^{ab} \bmod p$ given the two public values $g^a \bmod p$ and $g^b \bmod p$ when the prime p is sufficiently large. Breaking the Diffie-Hellman protocol is equivalent to computing discrete logarithms under certain assumptions.

10 Another system is the Point-to-Point Protocol (PPP) Extensible Authentication Protocol (EAP). EAP is a general system for PPP authentication that is compatible with a variety of authentication mechanisms. EAP does not select a specific authentication mechanism at a Link Control Phase, but rather postpones this selection until an Authentication Phase. This postponement enables the authenticator to request more information prior to 15 determining the specific authentication mechanism. In addition, this postponement also enables the use of a "back-end" server that actually implements the various mechanisms while the PPP authenticator merely passes through the authentication exchange.

RSA is yet another protocol system that provides an algorithm for public key 20 cryptography. The "key" of an RSA cipher has three numbers: the first is the public exponent, the second is the private exponent, and the third is the modulus. The public key is formed from the public exponent and the modulus. The private key is formed from the private exponent and modulus. If two devices are to engage in encrypted communications, they each generate a pair of keys. These devices then may exchange public keys using a non-secure

communications channel. Thereafter, when the devices engage in encrypted communications, one device can encrypt the message using the other devices' public key and send it via a non-secure channel. Since the private keys are not exchanged, decryption by an eves dropper proves difficult.

5 Consider the case of a wireless network with an access point in infrastructure mode. Suppose a user buys a wireless printer and wants to connect the printer to the network. If Wi-Fi Protected Access (WPA) is enabled on the access point, the user has a variety of options for setting up a secure connection between the access point and printer.

The user can install a WPA pre-shared key on the access point and on the printer.

10 Note that pre-shared keys are not device-specific. Also, multiple devices may utilize the same pre-shared key to connect to an access point. Alternatively, if the access point is a client to a Remote Authentication Dial in User Server (RADIUS), or includes the capabilities of a RADIUS server, the printer name and credentials can be added to the RADIUS server database. A RADIUS server is used to authenticate and return parameters including the users
15 IP address, gateway, and DNS server. The printer credentials must also be installed on the printer. The credentials may be a password, key, or certificate. The RADIUS server and printer are also configured to perform the same type of EAP authentication, with the printer acting as the supplicant.

Summary of the Invention

20 The present invention is for an Asynchronous Enhanced Shared Secret Provisioning Protocol (ESSPP). Asynchronous ESSPP provides a novel method and system for adding devices to a network in a secure manner. With Asynchronous ESSPP, at least one of two network devices that are attempting to register with each other and establish a secure

communications link is provided with a mechanism for starting Asynchronous ESSPP.

Examples of these network devices include servers, wireless printers, wireless computers, and network access points, but the protocol can be used for peer-to-peer and other devices. The mechanism may be to select a button located on the device that triggers the Asynchronous

5 ESSPP process, a menu selection provided in a Graphical User Interface (GUI) shown on a display provided with the device, or to enter a code on a terminal of the device, or other trigger mechanism. When, for example, two devices such as the printer and access point run Asynchronous ESSPP at the same time, the two devices automatically register with each other. The Asynchronous ESSPP process allows for registration of network devices without

10 the need to manually install keys, passwords, or certificates to add the device to the network.

With Asynchronous ESSPP, the registration process is activated by triggering a button or other trigger on the network device. The network device then initiates communications with the network to register itself. The registration is not activated until an out-of-band verification process is performed, *i.e.*, in a communications channel different from the one used for the in-band communications. If the verification process gives assurance that the registration was secure, the registration may be accepted. Once accepted, the network device and the network can use a shared secret created in the registration process for authentication or other cryptographic function such as generating a credential.

When a device runs Asynchronous ESSPP, it searches for another device running
20 Asynchronous ESSPP. When two devices running Asynchronous ESSPP detect each other they exchange identities and establish a shared secret that can later be used by the devices to mutually authenticate each other and generate session encryption keys. In a wireless network,

the shared secret can be used as a pre-shared key or it can be used for 802.1x authentication using an Extensible Authentication Protocol (EAP).

Brief Description of the Drawings

The present invention is further described in the detailed description which follows, in 5 reference to the noted drawings by way of non-limiting examples of certain embodiments of the present invention, in which like numerals represent like elements throughout the several views of the drawings, and wherein:

Figure 1 illustrates a Wi-Fi network that supports Asynchronous ESSPP in accordance with a preferred embodiment of the present invention;

10 Figure 2 illustrates a “man-in-the-middle” attack on a Wi-Fi network that supports Asynchronous ESSPP in accordance with a preferred embodiment of the present invention; and

Figure 3 illustrates a block diagram of an authentication server and a network device in accordance with a preferred embodiment of the present invention.

15 Detailed Description of Preferred Embodiments

The particulars shown here are by way of example and for purposes of illustrative discussion of the embodiments of the present invention only and are presented in the cause of providing what is believed to be the most useful and readily understood description of the principles and conceptual aspects of the present invention. In this regard, no attempt is made 20 to show structural details of the present invention in more detail than is necessary for the fundamental understanding of the present invention, the description taken with the drawings making apparent to those skilled in the art how the several forms of the present invention may be embodied in practice.

A novel method and system for registering devices with a network in a secure manner is provided through an Asynchronous Enhanced Secure Shared Provisioning Protocol (ESSPP). With Asynchronous ESSPP, at least one of two network devices that are attempting to register with each other and establish a secure communications link are provided with a 5 mechanism for starting Asynchronous ESSPP.

A preferred general process flow for Asynchronous ESSPP is provided in protocol flow 1 below. The mathematical terminology used for the purposes of describing the Asynchronous ESSPP process flows is documented in the Internet Engineering Task Force (IETF) Internet Draft “draft-moskowitz-shared-secret-provprotocol-01.txt”. This IETF 10 document describes a Shared Secret Provisioning Protocol (SSPP). Where possible, the same notation used in SSPP is used here.

(Xs, Ys)	Server's Diffie-Hellman static key pair
(Xc, Yc)	Client's Diffie-Hellman static key pair
(p, q, g)	Diffie-Hellman domain parameters
15 AddressS	Server's address or name
NonceS	Random number generated by Server used in the exchange
NonceSecurityCode	Random number generated by Server; used in calculation of the SecurityCode
NonceSecurityCodeHash	A hash of the NonceSecurityCode and NonceS. Calculated as:
20	$\text{SHA1}(\text{NonceS} \parallel \text{NonceSecurityCode})$
AddressC	Client's address or name
NonceC	Random number generated by Client used in the exchange

Zs	Diffie-Hellman generated shared secret ($Y_s^X_c \bmod p$) or $((Y_c^X_s \bmod p)$
kdf	Key Derivation Function specified in SSPP
k	Shared key generated by Client and Server as:
5	$kdf(Z_s, AddressC, AddressS, keydatalen, hashlen, NonceC, nonceS)$
proofS	A hash generated by Server to prove he knows Zs. Calculated as: $LTRUN96(HMAC-SHA1(Z_s, (AddressS \parallel NonceC \parallel NonceSecurityCode)))$
10	LTRUN96 performs a left truncation, returning the left most 96 bits of data.
proofC	A hash generated by Client to prove she knows Zs. Calculated as: $LTRUN96(HMAC-SHA1(Z_s, (Y_c \parallel AddressC \parallel NonceS)))$
SecurityCode	A hash generated by both client and server that may be manually inspected to ensure no man-in-the-middle participated in a registration.
15	Calculated as: $Base32Encode(LTRUN40(SHA1(Z_s \parallel NonceSecurityCode)))$

Protocol flow 1 below illustrates Asynchronous ESSPP registration of a network device with a network. Asynchronous ESSPP allows for the registration of the network 20 device with the network without a synchronous activation of triggers on both the network device and the network. With Asynchronous ESSPP, the registration process is activated by triggering a button on the network device. The network device then initiates communications with the network to register itself. The registration is not activated until an out-of-band

verification process is performed, *i.e.*, in a communications channel different from the one used for the in-band communications. If the verification process gives assurance that the registration was secure, the registration may be accepted. Once accepted, the network device and the network can use credentials created in the registration process for authentication.

5

Protocol Flow 1

Network

Network Device

1. Start Asynchronous ESSPP

2. Generate NonceS and send message

(p,q,g), Ys, AddressS, NonceS, NonceSecurityCodeHash

10

3. Validate parameters as required by SSPP, generate NonceC, Zs, proofC and send message.

Yc, AddressC, NonceC, proofC



4. Generate Zs, validate proofC,

15 generate proofS and send message

NonceSecurityCode, proofS

5. Validate NonceSecurityCode and proofS

6. Generate shared key k and

20 SecurityCode. k can be used to set up a secure connection between the network and the network device

6. Generate shared key k and

SecurityCode. k can be used to set up a secure connection between the network and the network device

<p>after the registration is accepted by the Network.</p> <p>7. Pause registration process</p> <p>8. Perform out-of-band signature verification</p> <p>5 by comparing Network SecurityCode with Network Device SecurityCode.</p> <p>Accept registration if codes match.</p> <p>9. Use shared key k to establish secure</p> <p>10 connection with Network Device.</p>	<p>after the registration is accepted by the Network Device.</p> <p>7. Pause registration process</p> <p>8. Perform out-of-band signature verification by comparing Network SecurityCode with Network Device SecurityCode. Accept Registration if codes match.</p> <p>9. Use shared key k to establish secure connection with Network.</p>
--	--

With Asynchronous ESSPP registration, a network device and network register with each other during a protocol exchange illustrated above in steps 1-6. The Asynchronous ESSPP registration is launched in step 1 at the network device. Subsequently in step 2, the network generates the values NonceS, NonceSecurityCode, and NonceSecurityCodeHash, and 15 transmits (p,q,g), Ys, AddressS, NonceS, and NonceSecurityCodeHash via a message to the network device. In step 3, the network device validates the parameters according to SSPP. In addition, the network device generates NonceC, Zs, proofC. Then, the network device transmits these values along with Yc to the network. Next in step 4, the network generates Zs, validates proofC, and generates proofS. The network then transmits NonceSecurityCode and 20 proofS to the network device. In step 5, the network device validates NonceSecurityCode and proofS. The network device validates NonceSecurityCode by using it to generate NonceSecurityCodeHash and making sure that it generates the same value for NonceSecurityCodeHash that was received from the network. After the protocol exchange in

steps 1-5, the network device and network generate SecurityCodes (signatures) or other information for testing identity of the ostensibly shared secret, without revealing the shared secret. In step 6, both the network and network device generate an ostensibly shared key k . After steps 1-6, both the network device and network have an ostensibly shared secret from 5 the registration process. In step 7, the registration process is paused pending an out-of-band verification that occurs in step 8. The shared secrets are the same if a man-in-the-middle has not compromised the communications between the network device and the network. If the SecurityCodes or other indicia of the ostensibly shared secret are determined to be the same, which occurs in step 8, an administrator can accept the registration of the network device with 10 the network, and a user can accept the registration of the network with the network device. The network device and network can then use registration credentials to authenticate each other.

The SecurityCode (signature) verification process in step 8 occurs in a communication channel that is different from the registration process. The signature verification process in 15 step 8 is therefore referred to as an out-of-band process. Unless the signature, or other indicia of the ostensibly shared secret, are validated, the registration of the network device with the network is not accepted. A network administrator may inspect registration information at the network device and at the network. The administrator may choose to accept the registration before the network device may use the registration credentials for authentication. A network 20 device may display, print, or otherwise communicate a signature or other information that the administrator may use to assure herself that the ostensibly shared secrets are the same. The administrator can choose to accept the registration if the signatures are the same. Similarly, the system administrator can choose to reject (delete) them if the registration signatures are

dissimilar. This validation process allows for the Asynchronous addition of new network devices to the secure network. The user of the network device can also choose to reject (delete) registration credentials if signatures are dissimilar; this protects the network device from connecting to an imposter network.

5 Preferably the ostensibly shared secrets are developed using any of a variety of protocols based on Diffie-Hellman key exchange. The Diffie-Hellman exchange protocol generates a shared secret. The registration signature used in the signature verification process of step 8 is derived from this shared secret.

10 There are a variety of ways of generating registration signatures used for the verification process of step 8. Registration signatures are preferably formed, in a non-limiting example, from a hash of secret information. For example, the registration signature may be formed from a hash of client public key, a hash of server public key, a hash of client/server public keys, a hash of client/server shared key, or a hash of client/server shared key and registration nonce.

15 A variety of indicia can be used to verify that the ostensibly shared secrets are the same. One indicia of the ostensibly shared secret may be a hash of the secret. Another indicia may be a value computed by exponentiating a base by the secret. If the network device and network have matching registration signatures, there is some assurance that there was no man-in-the-middle attacker during the key exchange protocol. This assurance is based on the fact 20 that when a man-in-the-middle attack is mounted on a Diffie-Hellman exchange, the participants do not generate the same secret value.

There are a variety of out-of-band channels for the validation of signatures for step 8. One method is by providing the network device with a ‘save’ feature that allows for the

saving of the registration signature and other information to a file on a transportable medium, such as a floppy disk. Such additional information may include a registered user name and associated registration signature. It is then possible to deliver the transportable medium to a system administrator. Another method is to provide the network device with a ‘print’ feature

5 that allows for the creation of a human readable hardcopy of signature information. Such information may include the registered user name and associated registration signature. It is then possible to use this printout in the signature validation process.

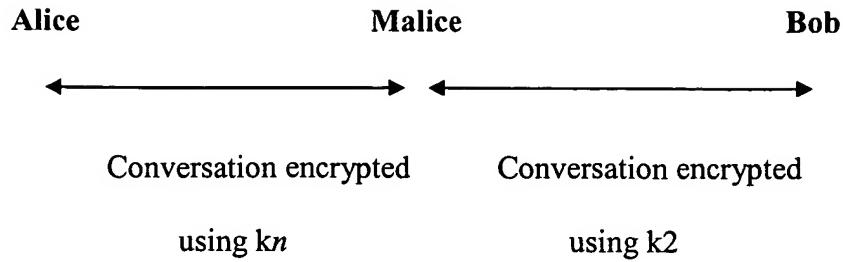
With Asynchronous ESSPP, the network device may determine a default name under which it can register. The network administrator may be given the option of overriding this

10 name, or the network administrator can use the default name and perform a registration without having to enter a name. There is also an optional generation of pseudonyms that can be used by the devices to identify each other during EAP authentication without revealing their true identities on the network. Asynchronous ESSPP also provides for an optional anonymous registration that allows two devices to run Asynchronous ESSPP without

15 revealing their identities on the network. In addition, with Asynchronous ESSPP, the operator of the network device need not enter a password or shared key, and need not install certificates. Credentials are established automatically during the registration process.

The out-of-band signature verification of protocol flow 1 provides security against a so-called man-in-the-middle attack. A man-in-the-middle attack occurs where an imposter

20 Malice attempts to interject himself between Alice and Bob who are trying to communicate with each other.



5 When Alice thinks she is talking to Bob, she in fact is talking to Malice, and when Bob thinks he is talking to Alice, he in fact is talking to Malice. The man-in-the-middle attack allows Malice to establish a shared secret k_n with Alice and a shared secret k_2 with Bob. If Malice can cause the signature derived from k_n to be the same as the signature derived from k_2 , then Malice can mount a successful attack.

10 Protocol flow 2 that follows demonstrates an exemplary process of how Malice could mount a man-in-the-middle attack when registration signatures are being used to protect a Diffie-Hellman exchange between Alice and Bob.

Flow Notation

	(Xa,Ya)	Alice's Diffie-Hellman key pair (Xa is private;Ya is public)
15	(Xb,Yb)	Bob's Diffie-Hellman key pair
	(Xm1,Ym1)	One of Malice's Diffie-Hellman key pairs
	(Xm2,Ym2)	Malice's Diffie-Hellman key pairs used in exchange with Bob
	(Xmn,Ymn)	Malice's Diffie-Hellman key pairs that allows Malice to perform a successful attack
20	kn	Malice and Alice's shared secret calculated as Xa^m or Xm^a
	k2	Malice and Bob's shared secret calculated as Xb^m or Xm^b
	rsn	Alice and Malice's registration signature derived from Alice and Malice's shared key as $\text{HASH}(kn)$

rs2

Malice and Bob's registration signature derived from Malice and
Bob's shared key as $\text{HASH}(k2)$

Protocol Flow 2

Alice

Malice

Bob

5

1. Alice sends Malice her public key

Y_a 

2. Malice sends Bob a public key

10

Y_{m2} 

3. Bob calculates $k2$ and $rs2$

and sends Malice his public key

 Y_b

15 4. Malice generates $k2$ and $rs2$. At this point in time,

Malice can try out a number of candidate key pairs

to send to Alice looking for a key pair that will

cause Alice and Malice's registration signature to

match $rs2$. Malice finds that using the key pair

20 (X_{mn}, Y_{mn}) results in a match, so Malice sends Y_{mn}

to Alice.

 Y_{mn}

5. Alice calculates k_n and rs_n

6. Alice and Bob do an out-of-band comparison of rsn and $rs2$ and find them to be equal. Malice has succeeded in the attack.

In a preferred process for registering a network device with a network, such as 5 protocol flow 1, the device and network use an in-band registration process that produces an ostensibly shared secret. Upon completion of an out-of-band verification that the ostensibly shared secrets are the same, a network administrator may choose to accept the registration process.

10 The use of this out-of-band verification process provides an ability to detect a man-in-the-middle attack. With Asynchronous ESSPP, Malice does not have enough information to generate SecurityCode registration signatures when he selects his Diffie-Hellman key pairs. As a result, if a man-in-the-middle attack does occur, there is a high probability that Malice 15 will not select key pairs that result in Alice and Bob's SecurityCodes matching. The out-of-band verification process illustrated in protocol flow 1 can reveal a SecurityCode mismatch and hence, detect his presence. Absent interference from Malice, $rs1$ will equal $rs2$, which is verified in the out-of-band process of protocol flow 1

If the verification process of step 8 occurred in the same communications band as the one used to generate the shared secret, Malice could continue to interject himself in the process and launch a man-in-the-middle attack during the verification process. However, by 20 using the out-of-band communications channel for verification, Malice is removed from the verification process. As a result, the use of the out-of-band verification process enhances the ability to detect the presence of Malice.

Minimizing Signature Length

It is possible to minimize the length of the registration signature while preserving the ability to detect the man-in-the-middle attack by modifying protocol flow 1 to include a commitment. Protocol flow 3 below demonstrates this.

5 Flow Notation

(Xa,Ya)	Alice's Diffie-Hellman key pair (Xa is private;Ya is public)
regNonceA	Registration nonce value generated by Alice.
hregNonceA	HASH(regNonceA)
regNonceM	Registration nonce value generated by Malice.
10 hregNonceM	HASH(regNonceM)
(Xb,Yb)	Bob's Diffie-Hellman key pair
(Xm1,Ym1)	Malice's Diffie-Hellman key pair used in exchange with Alice
(Xm2,Ym2)	Malice's Diffie-Hellman key pair used in exchange with Bob
k1	Malice and Alice's shared secret calculated as Xa^Ym1 or $Xm1^Ya$
15 k2	Malice and Bob's shared secret calculated as Xb^Ym2 or $Xm2^Yb$
rs1	Alice and Malice's registration signature derived from Alice and Malice's shared key as $\text{HASH}(k1 \text{regNonceA})$
rs2	Malice and Bob's registration signature derived from Malice and Bob's shared key as $\text{HASH}(k2 \text{regNonceM})$

Protocol Flow 3

Alice

Malice

Bob

1. Alice sends Malice her public key and a hash of a registration nonce

5 $Y_a, h_{regNonceA}$ 

2. Malice sends Bob a public key and a hash of a registration nonce

$Y_m2, h_{regNonceM}$ 

3. Bob calculates k_2 and sends Malice his public key

10  Y_b

4. Malice generates k_2 and rs_2 . At this point in time, Malice does not know how to calculate rs_1 , so Malice can only guess a public key that will cause rs_1 to equal rs_2 . Malice guesses Y_m1 and sends it to Alice.

15  Y_m1

5. Alice calculates k_1 and rs_1

Alice sends $regNonceA$ to Malice so that Malice can generate rs_1 .

$regNonceA$ 

- 20 6. Malice can validate $regNonceA$ by generating $h_{regNonceA}$ and verifying that it equals the value of $h_{regNonceA}$

previously received from Alice (but Malice, being
an attacker, may not care).

Malice generates rs1.

Malice sends regNonceM to Bob so that Bob can

5 generate rs2.

regNonceM →

7. Bob can validate regNonceM by generating hregNonceM

and verifying that it equals the value of hregNonceM

previously received from Malice. If the validation

10 fails, Bob detects that Malice did not stick to its

commitment to regNonceM and fails the exchange.

Bob generates rs2.

8. Alice and Bob do an out-of-band comparison of rs1 and

rs2 and are likely to find them not equal as Malice

15 was not able to try multiple candidate public keys

to find one that would result in rs1 equaling rs2.

In protocol flow 3, the nonce is an example of a commitment that allows the registration signature to be shorter than it otherwise would need to be to achieve a desired level of security. An additional advantage of the commitment is that it allows devices to re-
20 use key pairs without sacrificing security.

Registration signatures used to validate Diffie-Hellman key exchanges are currently long, often consisting of 20 byte SHA-1 hashes. A typical registration signature may be displayed as a string of characters consisting of hexadecimal digits such as:

F9 38 32 7B CE 15 58 44 78 35 97 74 88 F2 78 99 28 45 E9 AA

5 The example registration signature consists of 160 bits of information (4 bits per hexadecimal character). Generally speaking, a 160 bit signature is longer than is needed. It is possible to safely truncate a signature to 96 bits and provide good protection from third party attacks. However, it is desirable to reduce the registration signature size to something much lower than 96 bits in order to simplify the out-of-band signature examination. Also, it is
10 desirable to base32 encode the registration signature (encode the signature using lower case letters and the digits 2-7). This allows 5 bits of the signature to be encoded per character instead of 4. With EAP-Link, a typical registration signature looks like:

we9n-4pfk

This signature contains 40 bits of information, but because of protocol enhancements, still
15 allows for adequate protection from a man-in-the-middle attacker. With a 40 bit registration signature, Malice has a 1 in 2^{40} chance of guessing key pairs and a registration nonce value that causes rs1 to equal rs2.

A commitment increases the strength of the registration signature by an amount proportional to the number of key sets an attacker would otherwise be able to try out during or
20 prior to a Diffie-Hellman exchange. With Asynchronous ESSPP, an attacker can only try one set of keys and nonces during a registration exchange, thereby enhancing the security of the protocol exchange. When 40 bits of information are used for the signature as in the above

example, an attacker has a 1 in 2^{40} chance of providing keys and a registration nonce that will result in a successful attack with Asynchronous ESSPP. Security needs may vary with application and over time, so the length of a key needed to provide security may also vary. However, by use of a commitment, the length of key needed to provide a desired level of 5 security will be less than a key length for a substantially identical system that does not use a commitment.

Asynchronous ESSPP System Components

Figure 1 illustrates a Wi-Fi network 10 that supports Asynchronous ESSPP in accordance with a preferred embodiment of the present invention. Wi-Fi network 10 includes 10 network devices 12, 14, 16, and 18 that are numbered as devices 1, 2, n, and n+1 respectively. These numbered designations illustrate that network 10 may support as many as n+1 wireless devices.

Wireless network devices 12-18 are connected to Wi-Fi network through wireless access points 20 and 22 that are respectively numbered 1 and m. In order to support n+1 15 wireless network devices, network 10 may utilize as many as m wireless access points.

Wireless access points 20 and 22 numbered 1 and m may couple to other network devices 24, a virtual private network 26, as well as an authentication server 28. In addition, wireless access points 20 and 22 may couple to a terminal 30. Coupled to authentication server 28 is a storage facility 32.

20 Wireless network devices 12-18 are each provided with the mechanism that activates the Asynchronous ESSPP protocol on wireless network devices 12-18 as described in step 1 of protocol flow 1. This mechanism may take the form of a button, a switch, a menu selection on a Graphical User Interface (GUI), or some other embodiment of a trigger.

When Asynchronous ESSPP is activated on one of network devices 12-18 or authentication server 28, Asynchronous ESSPP proceeds to step 2 as outlined in protocol flow 1.

All Asynchronous ESSPP communications between wireless network devices 12-18 and authentication server 28 occurs through wireless access points 20-22. While network devices 12-18 are illustrated as communicating with authentication server 28 through wireless network 10, other forms of communication are possible, such as via satellite, dial up connection, broadband, cable, fire-wire, an Internet and World Wide Web, as well as other methods.

Figure 2 illustrates a “man-in-the-middle” attack on Wi-Fi network 10 that supports Asynchronous ESSPP in accordance with a preferred embodiment of the present invention. Depicted in Figure 2 is a man-in-the-middle device 40 that is attempting a man-in-the-middle attack on network 10. Man-in-the-middle device 40 broadcasts communications to both authentication server 28 or network device 18. Through broadcasting these communications, man-in-the-middle device 40 is attempting to interfere with the Asynchronous ESSPP process occurring between authentication server 28 and network device 18 and engage in a man in the middle attack. Through this man-in-the-middle attack, man-in-the-middle device 40 is attempting to register itself with authentication server 28 posing as wireless network device 18 and is also attempting to get wireless network device 18 to connect to it.

A block diagram depicting a system that supports the Asynchronous ESSPP is illustrated in Figure 3. Authentication server 28 supports a variety of subsystems that enable it to support the Asynchronous ESSPP. Among these systems are a key generator 40 that supports the cryptographic keys used with the Asynchronous ESSPP process. Specifically,

key generator 40 is able to create a cryptographic key in accordance with the Asynchronous ESSPP process, such as through a Diffie-Hellman process, an RSA process, or an EAP process.

Authentication sever 28 includes an internal database 44 and/or an external database 5 46. Databases 44 and 46 are provided with authentication server 28 to enable authentication server 28 to store information including network system addresses and identity information, pseudonym information, cryptographic key information, and other related information. A processor arrangement 50 is provided that couples to database 44 and 46, and key generator 40.

10 A communications system 52 is coupled to server that enables authentication server 28 to communicate with remote network devices 12-18 through wireless access points 20 and 22. Alternatively, other methods of communication are suitable including communication via satellite, a computer network such as the Internet and a World Wide Web, LAN, or Ethernet, and broadband.

15 A network device 60 seeking to securely register with authentication server 28 is also illustrated. As with authentication server 28, network device 60 also supports a variety of subsystems that enable it to support the Asynchronous Enhanced Shared Secret Provisioning Protocol. Network device 60 includes a registration trigger 62. This trigger 62 initiates execution of the Asynchronous ESSPP process on network device 60 when trigger 62 is 20 activated. Network device 60 includes a key system 64 that supports the cryptographic key in accordance with the Asynchronous ESSPP process. This support can include either generation or receipt and storage of a cryptographic key in accordance with the Asynchronous ESSPP process, such as through a Diffie-Hellman process, an RSA process, or an EAP

process. Also, it is possible that server 28 could receive the cryptographic key from network device 60.

Storage system 66 provided on network device 60 provides data storage for network address information, identity information, pseudo names, cryptographic key, and other related 5 information. A communications system 70 is provided to enable network device 60 to communicate with authentication server 28 through wireless or other communication methods. A processor arrangement 72 is provided that couples to storage system 66, trigger 62, communications system 70, and key system 64.

Diffie-Hellman Key Exchanges

10 The Diffie-Hellman key exchange is used to generate a key from which session keys can be derived. Wireless devices commonly use 128 bit session keys for symmetric encryption. The Diffie-Hellman key exchanges should therefore be as strong as a 128 bit symmetric key so that the Diffie-Hellman key exchange is not the weak link in the registration and authentication process.

15 The following table is taken from a recent Internet draft, "More MODP Diffie-Hellman groups for IKE". It is an equivalency table showing for different Diffie-Hellman prime moduli what symmetric key sizes are required to have equivalent strengths. The strength comparisons are estimates.

Group	Modulus	Strength Estimate 1		Strength Estimate 2	
		in bits	exponent size	in bits	exponent size
5	1536 bit	90	180	120	240
14	2048 bit	110	220	160	320
15	3072 bit	130	260	210	420
16	4096 bit	150	300	240	480
17	6144 bit	170	340	270	540
18	8192 bit	190	380	310	620

The table indicates that a 2048-bit Diffie-Hellman key provides the strength of a symmetric key probably somewhere between 110 and 160 bits in strength. Based on the estimates in this table, a 2048-bit Diffie-Hellman prime modulus is recommended for use with

5 Asynchronous ESSPP on wireless networks.

Asynchronous ESSPP may use prime moduli defined in the IETF draft, "More MODP Diffie-Hellman groups for IKE". In a wireless network, the Authentication Server may indicate, by group number, which Diffie-Hellman prime modulus may be used. Asynchronous ESSPP devices supports groups 5, 14, and 15.

10 Wei Dai has published benchmarks for Diffie-Hellman key exchanges at:<http://www.eskimo.com/~weidai/benchmarks.html>

His measurements indicate that Diffie-Hellman 2048 Key-Pair Generation and Key Agreement takes 49.19 milliseconds on a Celeron 850MHz processor running Windows 2000 SP1.

15 The Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed by Diffie and Hellman in 1976 and published in the ground-breaking paper

"New Directions in Cryptography." The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

The protocol has two system parameters p and g . They are both public and may be used by all the users in a system. Parameter p is a prime number and parameter g (usually 5 called a generator) is an integer less than p , with the following property: for every number n between 1 and $p-1$ inclusive, there is a power k of g such that $n = g^k \pmod{p}$.

Suppose Alice and Bob want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They proceed as follows: First, Alice generates a random private value a and Bob generates a random private value b . Both a and b are drawn from the set of 10 integers $\{1, \dots, p-2\}$. Then they derive their public values using parameters p and g and their private values. Alice's public value is $g^a \pmod{p}$ and Bob's public value is $g^b \pmod{p}$. They then exchange their public values. Finally, Alice computes $g^{ab} = (g^b)^a \pmod{p}$, and Bob computes $g^{ba} = (g^a)^b \pmod{p}$. Since $g^{ab} = g^{ba} = k$, Alice and Bob now have a shared secret key k .

The protocol depends on the discrete logarithm problem for its security. It assumes 15 that it is computationally infeasible to calculate the shared secret key $k = g^{ab} \pmod{p}$ given the two public values $g^a \pmod{p}$ and $g^b \pmod{p}$ when the prime p is sufficiently large. Maurer has shown that breaking the Diffie-Hellman protocol is equivalent to computing discrete logarithms under certain assumptions.

The Diffie-Hellman key exchange is vulnerable to a man-in-the-middle attack. In this 20 attack, an opponent Carol intercepts Alice's public value and sends her own public value to Bob. When Bob transmits his public value, Carol substitutes it with her own and sends it to Alice. Carol and Alice thus agree on one shared key and Carol and Bob agree on another shared key. After this exchange, Carol simply decrypts any messages sent out by Alice or

Bob, and then reads and possibly modifies them before re-encrypting with the appropriate key and transmitting them to the other party. This vulnerability is present because Diffie-Hellman key exchange does not authenticate the participants. Possible solutions include the use of digital signatures and other protocol variants.

5 It is noted that the foregoing examples have been provided merely for the purpose of explanation and are in no way to be construed as limiting of the present invention. While the present invention has been described with reference to certain embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of limitation. Changes may be made, within the purview of the disclosure, as 10 presently stated and as amended, without departing from the scope and spirit of the present invention in its aspects. Although the present invention has been described herein with reference to particular means, materials and embodiments, the present invention is not intended to be limited to the particulars disclosed herein; rather, the present invention extends to all functionally equivalent structures, methods and uses.